Список литературы

1. Гергель В.П. Теория и практика параллельных вычислений. – М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. – 424 с.
2. Сайт алгоритмического языка Chapel – http://chapel.cray.com.
3. Сайт алгоритмического языкаX10 – http://x10-lang.org.
4. Сайт алгоритмического языкаCAF – http://www.co-array. org.

В.П. Гергель, А.А. Сиднев

Нижегородский государственный университет им. Н.И. Лобачевского

О ПРИМЕНЕНИИ МАКРОМОДУЛЬНОЙ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММ

Разработка программного обеспечения является трудоемкой профессиональной деятельностью. Для упрощения разработки программного обеспечения разработчики выполняют декомпозицию задачи на более мелкие и простые подзадачи, организуя их в отдельные вычислительные блоки (модули). Разработанные модули могут быть использованы в дальнейшем повторно, оформляясь в виде отдельных библиотек. Модульный подход разработки программного обеспечения является на данный момент общепризнанным подходом.

Одним из недостатков такого подхода является отсутствие стандартов на интерфейсы модулей. Разработчик библиотеки сам определяет удобные ему структуры хранения данных и интерфейсы функций, которые их обрабатывают. В результате каждая библиотека получается уникальной и возникает сложность, связанная с заменой используемых библиотек.

Другая проблема при использовании библиотек – проблема выбора наиболее оптимальной библиотеки под текущие задачи проекта. Для решения широкого круга задач существует достаточно много библиотек. Каждая библиотека может иметь

свою сложность внедрения, эффективность реализации, удобство использования структур данных и поддержку различных программно-аппаратных платформ. Задача выбора лучшей библиотеки может не иметь однозначного решения, поэтому разработчикам приходится идти на компромисс при выборе библиотеки, теряя в эффективности реализации, удобстве использования или количестве поддерживаемых программно-аппаратных платформ. Со временем возможностей текущей библиотеки под решение очередной задачи может не хватать, поэтому разработчики могут столкнуться с задачей перехода на другую библиотеку в связи с такими причинами, как:

- переход на другую программно-аппаратную платформу;
- требуование большой эффективности;
- отказ от использования текущей версии библиотеки в связи с прекращением её поддержки (она устарела).

При переходе на новую библиотеку разработчику придётся столкнуться с необходимостью выполнить модификацию используемых структур данных и функций под те, которые используются в библиотеке. Такой переход может быть очень трудоёмким.

Рассматриваемая макромодульная технология (ММТ) разработки программ позволят решить указанные проблемы.

Идея технологии. В основе технологии макромодульной разработки программ лежит идея макроописания участков кода программы. Такое описание содержит информацию о том, какие действия выполняются в указанном участке кода и какие структуры данных при этом используются. Действием может быть решение системы линейных уравнений, перемножение матриц, выполнение быстрого преобразования Фурье и др.

На основании описания макросреда, реализующая поддержку макромодульной технологии, может выполнять замену пользовательского кода на вызовы библиотечных функций. Каждое макроописание соответствует абстрактной функции, а для каждой абстрактной функции может существовать несколько реализаций из одной или нескольких библиотек (рис. 1). Таким образом, разработчик получает возможность использовать под-

ходящую ему библиотеку без внесения дополнительных изменений в код. Макросреда берёт на себя работы, связанные с сопряжением интерфейсов, используемых в пользовательской программе и целевой библиотеке.

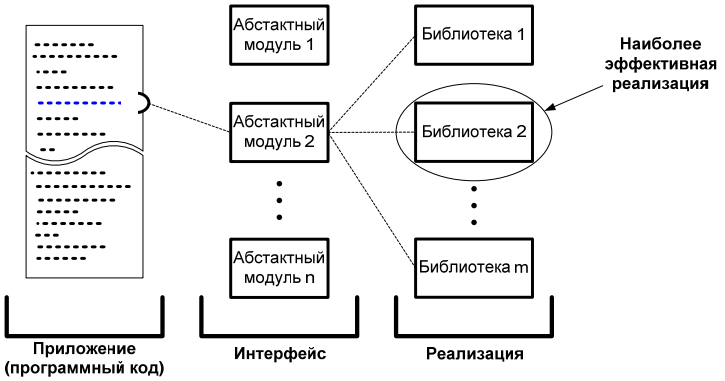

Рис. 1. Общая схема макромодульной технологии

Реализация технологии. Технология макромодульной разработки программ может применяться к любому языку программирования. Далее будет рассматриваться язык C/C++.

Наиболее удобным средством для выполнения макроописания в языке C/C++ являются директивы препроцессора [1]. Мы будем использовать директивы pragma, которые предназначены для реализации возможностей компиляторов и наилучшим образом подходят для выполнения макроописания. В том случае, если среда разработки не поддерживает макроописания, соответствующие директивы будут проигнорированы и будет собрано приложение, написанное пользователем.

Каждое макроописание соответствует абстрактной функции с теми типами данных, которые используются в программе пользователя – метафункции. Абстрактная функция может иметь несколько реализаций из одной или нескольких библиотек. Выбор наиболее подходящей реализации может быть выполнен пользователем.

Для упрощения модификации, использования и расширения возможностей макромодульной технологии все необходимые данные хранятся в текстовых файлах формата расширяемого языка разметки XML. Для описания макросреды используется набор из 5 файлов (рис. 2):

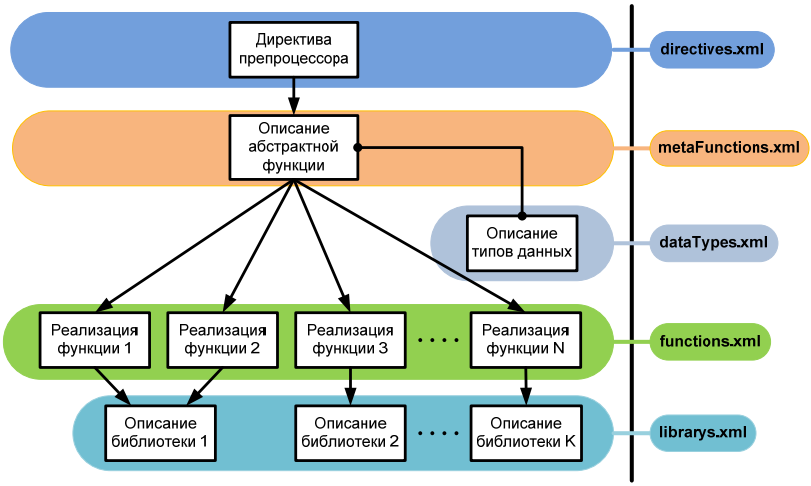

Рис. 2. Реализация макромодульной технологии

- описание поддерживаемых директив (directives.xml);
- описание метафункций (metaFunctions.xml);
- описание структур данных (dataTypes.xml);
- реализация функций (functions.xml);
- описание библиотек (librarys.xml).

Для каждой директивы создаётся запись в файле directives.xml с указанием абстрактной функции, которая выполняет указанный с помощью директивы код. Интерфейс функции описывается с помощью типов данных, представленных в dataTypes.xml. Доступные абстрактные функции представлены в файле metaFunctions.xml. Каждой абстрактной функции соответствует одна или несколько реализаций, использующих библиотеки. Описание таких реализаций представлено в functions.xml. Для каждой реализации указы-

ваются требуемые заголовочные файлы и статические библиотеки. Описание используемых библиотек содержится в librarys.xml. В описании указаны пути к заголовочным файлам, статическим библиотекам и динамическим библиотекам (эти пути могут задаваться при установке соответствующей библиотеки).

Использование макромодульной технологии происходит перед компиляцией исходных кодов программы на этапе препроцессирования. Для каждой директивы макромодульной технологии выполняется замена указанного блока в программном коде на соответствующую реализацию из библиотеки. Если таких реализаций несколько, то выбор функции осуществляется в соответствии с конфигурацией, указанной разработчиком.

Пример использования. Для удобства использования макромодульной технологии было разработано расширение к среде разработки Microsoft Visual Studio 2008, которое позволяет автоматизировать применении технологии [5]. Расширение позволяет выполнять предварительное препроцессирование исходных кодов программы и задавать целевую конфигурацию для сборки.

В качестве примера использования технологии была выбрана задача матричного умножения [2]. Программа содержит макроописание и простейшую реализацию матричного умножения. В качестве библиотечных реализаций были выбраны библиотека MKL [3] и технология для вычислений на графических картах CUDA [4].

Ниже представлены результаты вычислительных экспериментов на тестовой системе с видеокартой GeForce 8800 GTS и двухъядерным процессором Intel Core 2 E6650.

Из результатов эксперимента видно, что реализация пользователя неэффективна и выполнение матричного умножения на графической плате эффективнее, чем на процессоре на данной тестовой системе. На системе с большим количеством вычислительных ядер ситуация может измениться. Таким образом, пользователь получает три реализации программы с минимальными

затратами на разработку. Эффективность программы и используемой библиотеки определяется целевой платформой.

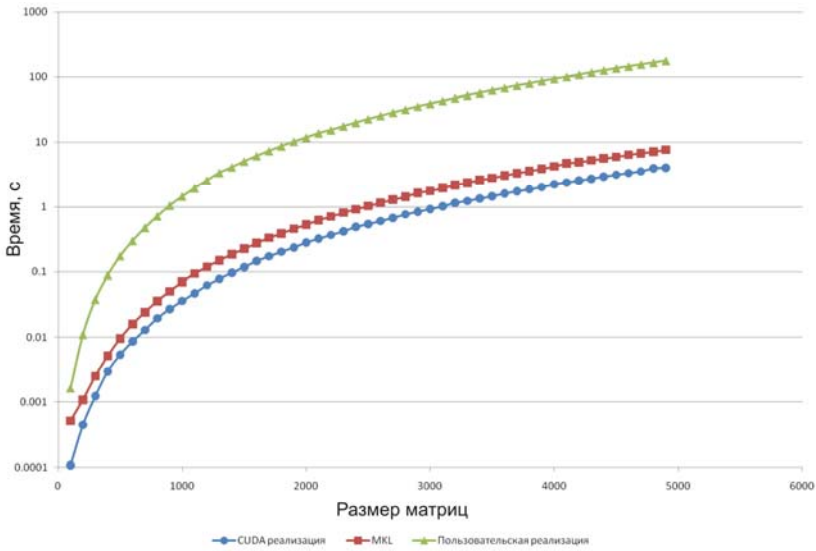

Рис. 3. Результаты экспериментов

Рассмотренная технология макромодульной разработки программ позволяет упростить разработку программного обеспечения за счёт сокращения затрат на внедрение библиотек. Пользователю достаточно сделать макроописание участков кода, и сборка приложения будет осуществляться автоматически с использованием наиболее подходящей библиотеки. В результате решается проблема с выбором наиболее оптимальной библиотеки, а переход на использование новой библиотеки значительно упрощается.

Для поддержки макромодульной технологии используется механизм директив препроцессора. В том случае если среда разработки не поддерживает макроописания, то соответствующие директивы будут игнорироваться и будет собрано приложение, написанное разработчиком.

Список литературы

1. Павловская Т.А. C/C++. Программирование на языке высокого уровня. – СПб.: Питер, 2003. – 461 с.

2. Knuth D.E. The Art of Computer Programming. Vol. 2: Seminumerical Algorithms. Addison-Wesley Professional; 3 edition. November 14, 1997. – P. 501.

3. Intel® Math Kernel Library. Reference Manual. September 2007.

4. NVidia CUDA. Reference Manual. Version 2.3. July 2009.

5. Craig S., Marc Y., Brian J. Working with Microsoft® Visual Studio® 2005. Microsoft Press.

В.П. Гергель, К.А. Чабан

Нижегородский государственный университет им. Н.И. Лобачевского

СИСТЕМА РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ DCS

В настоящее время многие организации имеют в своем распоряжении довольно крупную компьютерную сеть, объединяющую десятки и даже сотни компьютеров. С ростом такой сети все актуальнее становится вопрос об эффективности использования вычислительных ресурсов, которые она объединяет. Зачастую имеющиеся компьютеры загружены не полностью. Поэтому вполне естественно задействовать простаивающие ресурсы для выполнения какой-либо полезной работы – например, использовать их для решения некоторой трудоемкой задачи. Развитие данного направления привело к возникновению концепции *utility computing*.

Основная идея данной концепции состоит в использовании независимых компьютеров, объединенных в сеть, в качестве единой параллельной машины или виртуального суперкомпьютера. Эта идея имеет ряд преимуществ, например низкая стоимость суммарных вычислительных ресурсов.

Быстрый рост сети Интернет позволяет применить эти концепции в намного большем масштабе. К тому же настольные